Optimizing Real-Time Video Experience with Data Scalable Codec

Hanchen Li*
University of Chicago

Yihua Cheng*
University of Chicago

Ziyi Zhang
University of Chicago

Qizheng Zhang
Stanford University

Anton Arapin
University of Chicago

Nick Feamster
University of Chicago

Amrita Mazumdar
NVIDIA

ABSTRACT

Real-time video communication is becoming more and more important. However, packet loss is prevalent and resending packets, especially in long-latency networks, causes visual stalls. Previous solutions all perform suboptimally as they either add redundancy before sending the data, which *reduces bitrate* when no packet is lost, or fail to prevent *video freeze* when redundancy is not enough. User studies confirm that both bitrate decrease and video freeze significantly damage users' Quality of Experience (QoE). Through a user study comparing different artifacts during a quality drop period, we find that moderate quality drop is preferred over video freeze during packet loss. Inspired by this, we propose a new solution that trains a neural network Autoencoder to optimize frame quality under different packet loss rates. Our insight is that such training produces a *Data Scalable* codec, whose quality increases with each new packet arrival and reaches highest quality when no packet is lost. Specifically, with the arrival of any x encoded bytes of a frame, the decoded quality is closer to the quality than if the whole frame were encoded with x bytes in the first place. Thus, unless all packets are lost, our approach causes a moderate quality drop instead of video freeze during packet loss. In the end, we identify the technical challenges remaining in this approach and point out future opportunities.

CCS CONCEPTS

• **Information systems** → **Multimedia information systems**; • **Networks** → **Application layer protocols**; • **Computing methodologies** → *Computer vision*; • **Human-centered computing**;

KEYWORDS

Real-Time Communication, Video Conferencing, Autoencoder, QoE

ACM Reference Format:
Hanchen Li*, Yihua Cheng*, Ziyi Zhang, Qizheng Zhang, Anton Arapin, Nick Feamster, and Amrita Mazumdar. 2023. Optimizing Real-Time Video Experience with Data Scalable Codec. In *Emerging Multimedia Systems (EMS '23), September 10, 2023, New York, NY, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3609395.3611108

EMS '23, September 10, 2023, New York, NY, USA

ACM ISBN 979-8-4007-0303-4/23/09…$15.00
https://doi.org/10.1145/3609395.3611108

1 INTRODUCTION

Real-time video communication has become increasingly important [18]. With applications in online conferencing [5], cloud gaming [4] and virtual reality [8], real-time video is growing more rapidly than other video formats.

However, since there is little buffering for streaming real-time videos [15], they are more vulnerable to unstable network conditions. With Adaptive Bitrate Algorithms (ABR), the sender tries to align sending rate with bandwidth, but sudden network degradation such as a bandwidth drop could still cause packet loss [18]. Since the receiver does not keep a buffer in advance, it may need to wait for packet retransmission and this damages video quality.

There has been a wide range of effort spent to solve this problem: Forward Error Correction (FEC) adds redundancy to the packets such that the transmitted data could recover from packet loss. Error Concealment methods decode partial frames and try to recover the original frames. Scalable Video Coding (SVC) encodes videos at multiple quality layers, enabling incremental improvement of quality as data is transmitted layer by layer.

However, none of these methods achieve satisfactory performance for both packet loss and non-loss situations. FEC sets a fixed redundancy rate. But if the predicted loss rate is lower than the actual loss rate, the data cannot be reconstructed. On the other hand, a higher predicted loss rate leads to streaming video at a lower bitrate, compromising the video quality. Error concealment techniques give lower quality when streaming without loss and suffer from substantial degradation in the presence of packet loss. SVC assumes packets are transmitted in layer order. If a packet in lower layer is lost, higher layers will not be able to decode. This gives lower quality and could still need retransmission.

Prior work has shown that bitrate drop and frame freeze both damage Quality of Experience (QoE). [14, 17] In order to specifically quantify the tradeoff between them to find out properties that an ideal real-time video codec should satisfy, we conducted a supplementary user study in §2. It is observed that users prefer moderate graphical quality drops over frame freezes during packet loss, but this preference disappears when graphical quality drops becomes more severe.

Inspired by the observation, we propose the solution of Data Scalable codec to perform under both packet-loss and non-loss

*equal contribution

situations. By training neural-network based video autoencoders that optimize performance under different packet loss rates (§3), we create the first *Data Scalable* codec that exhibits a *progressive improvement* in quality with *any* more data arrival. Specifically, when 100% of the packets arrive, the codec delivers video quality comparable to traditional codecs such as H.265 in the same bitrate. With any of 60% data arrival, it maintains a quality similar to H.265 encoding with 60% bitrate beforehand. While it gives high quality when there is no packet loss, this Data Scalable codec reduces the need for retransmission and substitutes it with moderate quality drop under packet loss. No previous methods have achieved this property: FEC doesn't increase quality when more data arrives after being able to reconstruct and can not decode if not enough arrives. SVC forces packets to arrive in an enforced order. Error concealment solely modifies the decoder and do not produce comparable quality since much information has already been lost.

With Data Scalability, our codec has the potential to bring great improvement to real-time communication. Through end to end simulation, we show that, compared with existing systems, our solution reduces the ratio of delayed frames (defined as the frames whose delay between encoding and decoding is greater than 400 ms) by up-to 19× and the stall rate (defined as the fraction of video freezes that exceed 200 ms) by 14× with comparable visual quality.

To summarize, our contributions in this paper are the following:

- We point out that an ideal codec that performs under both packet loss and no loss conditions should be Data Scalable to deliver quality proportional to the amount of received data.
- We built the first Data Scalable codec by modifying neural network based autoencoder that has significant potential improvement over existing systems.
- We identify challenges and future opportunities brought by this idea to spark more effort in the direction

2 MOTIVATION

2.1 Real-Time Video Communication

Real-time video communication can be peer-to-peer or with media servers [3, 18]. Although the technique applies to more general situations, this paper mainly focuses on the transmission of peer-to-peer video. In such cases, the video sender runs a rate adaptation logic to dynamically set the frame rate and bitrate of the encoded video stream. The video stream consists of multiple groups of consecutive frames (GoP), each starting with an I-frame (intra-frame) followed by P-frames (predicted frames). I-frames are encoded independently (*i.e.,* they can be decoded without other frames), so their sizes are much bigger than P-frames, which only encode the differences with other *reference* frames. The key difference from traditional video on demand (VoD) streaming is that there is little buffer at the receiver since the content needs to be delivered as soon as possible [10, 15]. This makes real-time video especially vulnerable to packet loss in dynamic networking conditions.

2.2 Previous Methods

Many efforts have been done to mitigate packet loss for video streaming. We discuss common previous approaches here.

Forward Error Correction coding (*e.g.,* FEC): In real-time video communication, the most commonly used technique for dealing with packet losses is forward error correction (FEC) [10, 30]. FEC encodes an n-byte frame to (n+k) bytes such that when any n of the (n+k) bytes arrive at the receiver side, it is able to reconstruct the original data. The problem with FEC-based methods is that it requires prior knowledge of the packet loss rate. Without this knowledge, the effective bitrate is reduced or retransmission may still be necessary. If the amount of redundancy is underestimated, the original frame still cannot be reconstructed. This results in retransmitting the lost packets, which causes video freeze. Conversely, if sender overestimates the loss rate to be higher than the necessary redundancy rate, it degrades the video quality.

Error Concealment: There has been work trying to reconstruct the original frame based on partially decoded frames. H264 has inbuilt error concealment methods [22]. Recent studies have explored the use of neural networks (NN) to recover motion vectors (MVs) or residuals [19]. However, these error concealment techniques adds redundancy to data since different areas (macroblocks) of a frame need to be independently decodable. This causes quality drop during non-loss scenarios. Moreover, as our evaluation results show in §4, their performance under packet loss is also suboptimal since the majority of processing is solely on decoder side.

Scalable Video Coding (*e.g.,* SVC): SVC divides a video into multiple quality layers and transmits them incrementally, allowing each newly received packets or layers to enhance the quality gradually. Notable advancements have been made in SVC-based techniques for on-demand video streaming including fine-grained scalability (FSG) and Swift [13, 23].

However, SVC enforces a layer order for the packets of a frame such that the loss of a single packet in lower layer can impact all higher quality layers and may cause retransmission. While it is possible to reliably transmit each layer individually in on-demand videos, real-time video clients typically send each frame as a burst to minimize delay. Therefore, without knowledge of the sequence of lost packet, SVC has undesirable performance under loss. It could add redundancy with FEC, but the problem of FEC still remains.

2.3 Quality of Experience

Given these problems, we try to quantify their impact to real world users and find a more ideal solution to improve experience.

There has been lots of research effort done in the area of Quality of Experience to align experience with a objective metric: Streaming QoE Index builds their QoE model as a linear addition of perceptual graphical quality and stalling score [14]; Seufert et al. compared different QoE models in crowdsourced video datasets to show that different models produce different rankings and it is hard to tell the correctness of model without a definite groundtruth score [32]; Plenty other models have included graphical quality/bitrate and temporal factors including stalling and buffering time as key indicators for QoE [11, 12, 15, 17, 29, 35].

Although there is enough evidence showing that systems should deliver videos with higher graphical quality while minimizing video freeze, there hasn't been an explicit study on the tradeoff between quality decrease and video stall in a quality-impairment event caused by packet loss. The closest study was done by Salsify in appendix [15]. However the bitrate decrease is applied to the whole

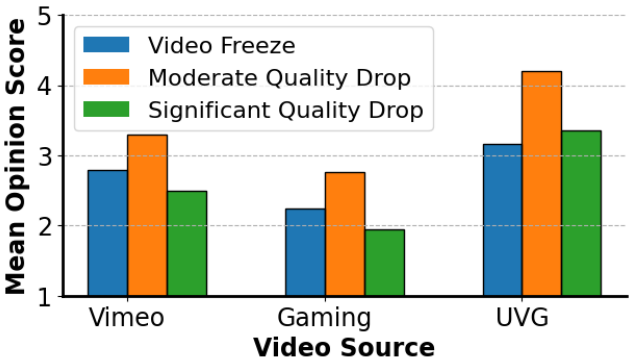


Figure 1: *User Study shows that users prefer moderate quality loss over video freeze during packet loss*

video chunk instead of the specific section where the packets get lost, and there is no real-world explanation for the graphical quality and delay level chosen in the study.

In order to explicitly learn about users' preference on the tradeoff of delay and graphical quality drop during packet loss in context of a streaming system, we conducted a user study using three 10-second reference videos from three different sources: cloud gaming [4], Vimeo [36], and codec analysis[2]. We encode the source videos in 2Mbps normally and simulate different quality impairment effects in packet loss situations where there is sudden bandwidth drop to 500Kbps in 5th and 7th second. The impairment effects created are the following: (1) ~800ms video freeze (generated by simulator in §4.2) (2) Modest graphical quality drops (by encoding with 500Kbps during bandwidth drop) 3. Significant graphical quality drop (by encoding with 200Kbps during bandwidth drop).

Through 135 user ratings collected with the crowdsourcing website Amazon Mturk [6], we observe that users prefer modest quality drop over video freeze with 25.3% higher average Mean Opinion Score. This confirms that video freeze indeed needs to be prevented. Users prefer to keep the video playing even with degraded content. However, when visual quality drops by too much, this preference over video freeze disappears. When the frame becomes too obfuscated and users fail to recognize the content, they would rather wait for clear frames to come later. Moreover, notice that the moderate quality drop video generated by 500Kbps biterate is equivalent to streaming with an oracle bitrate controller that knows future bandwidth condition. Thus the quality should be optimal under such packet loss scenarios.

By explicitly comparing effects on QoE of these two potential quality impairment events under packet loss and combining with previous studies, we are the first to point out an ideal real-time video codec that enables better experience under both packet-loss and non-loss situations should have the following properties:

- It is able to encode and decode in highest quality when there is no packet loss.
- When there is packet loss, it needs to prevent both significant graphical quality damage and video freeze. On the other hand, a modest quality drop is acceptable.

2.4 Data Scalable Codec

We condense the above properties into a formal definition of *Data Scalability*:

> When x% of the encoded N-byte frame arrives, the decoded quality of this frame is almost as good as encoding $\frac{Nx}{100}$ bytes and decoding with full data.

With Data Scalable Codecs, users will be able to receive full quality frames when there is no packet loss. When there is packet loss, Data-Scalable codecs approximate the quality of frame encoded with real arrived frame size. For example, in the setting of the second user study experiment, Data Scalable codec will give frame quality close to the optimal quality that was encoded in 500Kbps during bandwidth drop period, which is similar to sending with a oracle bitrate controller. And this will give us the preferred moderate quality drop during packet loss. Since its quality with full data arrival is also close with normal codec, it overcomes the problem of lowered quality in adding redundancy.

Interestingly, a Data Scalable codec does not need full prediction of network condition such as future bandwidth or loss rate. Rather, it tries to make best use of every packet arrival such that each new arrival increases frame quality. This is particularly helpful for one-on-one video communication facing dynamic network conditions.

3 NEURAL CODEC FOR DATA SCALABLILITY

We built the first Data Scalable codec for real time video communication by revamping existing neural network based autoencoder structure that purely optimized for video compression [25]. We will first introduce Autoencoder video codec and then explain our Data Scalable customization.

3.1 Autoencoder Structure

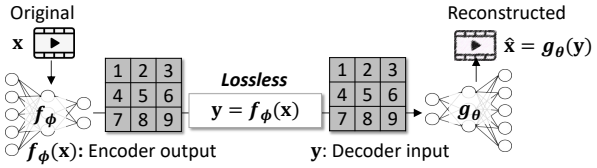
Unlike traditional codecs that uses handcraft logic, Autoencoders utilize learned neural networks (NNs) to work as encoders and decoders. We denote the encoder by f_ϕ (with its NN weights as ϕ) and the decoder by g_θ (with its weights as θ). The encoder encodes a frame $\mathbf{x}$ to a *coded tensor* $\mathbf{y} = f_\phi(\mathbf{x})$, and the decoder decodes the coded tensor $\mathbf{y}$ to a reconstructed frame $\hat{\mathbf{x}} = g_\theta(\mathbf{y})$.

Mathematically, a video compression autoencoder typically minimizes the following expected loss function during training:

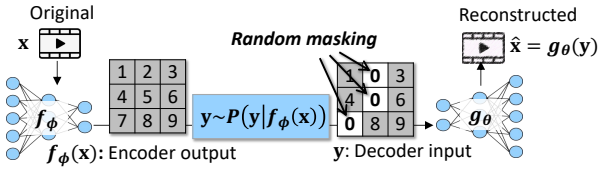
$$\mathbb{E}_{\mathbf{x}} \underbrace{Err(g_\theta(\mathbf{y}), \mathbf{x})}_{\text{Pixel error}} + \alpha \cdot \underbrace{Size(\mathbf{y})}_{\text{Encoded size}}, \text{ where } \mathbf{y} = f_\phi(\mathbf{x}) \quad (1)$$

Here, $Err(\hat{\mathbf{x}}, \mathbf{x})$ is the graphical pixel-level error of the decoded frame $\hat{\mathbf{x}}$ compared with original frame, and $Size(\mathbf{y})$ is the size of $\mathbf{y}$ in bit-per-pixel (bpp). The parameter α determines the average size generated by a model: a higher α leads to smaller frame size ($Size(\hat{x})$). But it would also degrades the graphical quality of $\hat{x}$ since the frame error term has less weight in the loss function. This enables autoencoder codecs to satisfy different bitrate requirement.

This work focuses mainly on encoding P-frames, the most common frame type in real-time videos as mentioned in §2. Specifically, we use the architectures of DVC [25], a popular video autoencoder. It shares similar steps with traditional video codec that it has motion estimation, motion compensation, residual encoding/decoding, and



(a) **Standard autoencoder training:** Lossless channel between the NN encoder and the NN decoder.



(b) **New autoencoder training:** Random masking on the encoder output to simulate the effect of packet loss.

Figure 2: *We mask random fractions of elements in the coded tensor (encoder output) to zero in contrast to original autoencoder.*

entropy coding. However, it replaced handcrafted heuristics in these steps with neural networks. Since every function is differentiable, every step could be trained jointly to minimize the loss function in Eq. 1. This end-to-end training gives it close performance with H265 under various bpps.

3.2 Data Scalable Customization

We now present our modifications to video compression autoencoders that enable Data Scalablity. Instead of minimizing previous loss funcition in Eq 1, we jointly train the encoder and decoder NNs to minimize:

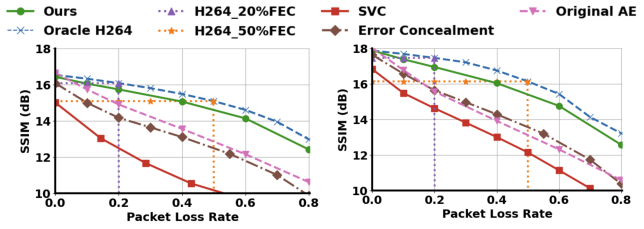
$$\mathbb{E}_{\mathbf{x}} Err(g_\theta(\mathbf{y}), \mathbf{x}) + \alpha \cdot Size(\mathbf{y}), \text{ where } \underbrace{\mathbf{y} \sim P(\mathbf{y}|f_\phi(\mathbf{x}))}_{\text{Simulate packet loss}} \quad (2)$$

The essential part of the loss function in Eq. 2 is the distribution function P (highlighted in blue), which describes the distribution of coded tensor after applying loss. With a packet loss rate of k%, $P(\mathbf{y}|f_\phi(\mathbf{x}))$ is the probability of $\mathbf{y}$ being the result of randomly masking k% of elements of the coded tensor $f_\phi(\mathbf{x})$ to zero.

Since during majority of the time a video stream does not suffer from packet loss, we set the distribution to having 0 loss in 80% of the time. In the other 20% of time, $P(\mathbf{y}|f_\phi(\mathbf{x}))$ is set to a uniformly random choice in {10%,20%,...,60%} to simulate different levels of loss. This setup enables the codec to handle different values of packet loss rate and maintain high quality during majority of streaming.

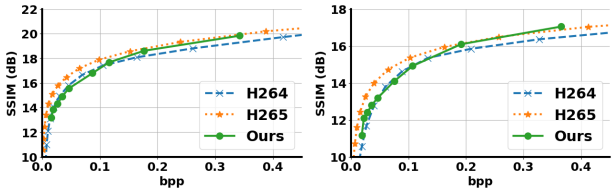
Note that the probability distribution P may not be expressed as an explicit function of ϕ and is thus not differentiable. To train the weights end to end, we use the Reinforce trick [21] to approximate gradient by Monte-Carlo Sampling.

Moreover, for each packet to be separately entropy decodable, the distribution information in entropy coding needs to be copied inside every packet, causing up to 40% overhead. We solved this problem by utilizing the trainability of neural networks. By adding regularization during training to make each channel of the coded tensor follow a zero-mean Laplace distribution, we can describe the distribution of each channel solely by their variance. Since there



(a) bpp = 0.1 (b) bpp = 0.2

Figure 3: *Data Scalable Codec performs close to Oracle H264 that knows loss rate beforehand and sets redundancy accordingly*



(a) FVC Dataset (b) Cloud Gaming Dataset

Figure 4: *Our Data Scalable Codec has comparable coding efficiency in different dataset and bpp*

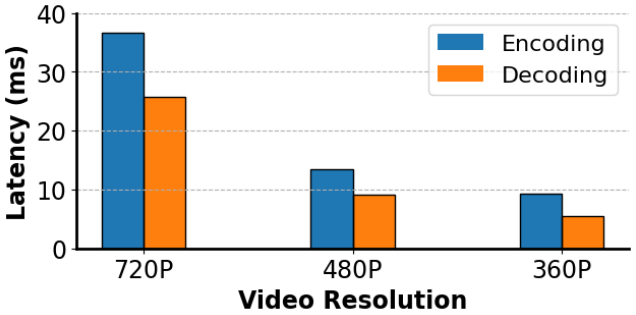


Figure 5: *Data Scalable codec runs with decent speed on different resolution videos with NVIDIA A40 GPU*

are only 224 channels in DVC's coded tensor, this gives a less than 5% overhead. This enables us to treat each packet loss as randomly zeroing out the corresponding proportion of the coded tensor while not inflating the size by too much.

4 PRELIMINARY RESULTS

4.1 Setup

Training: We trained our Data Scalable autoencoder by fine-tuning from DVC model [25] on Vimeo-90K [36] dataset with generic content. Learning rate was set to 10^{-4} and training for a new model takes 3-4 hours with NVidia RTX 3080 GPU.

Test videos: We used 12 videos randomly sampled from two datasets of real-time videos (Cloud Gaming [4] and Video Conferencing [5]). Notice that none of the videos were in the training set. This prevents the overfitting problem.

Testbed and network traces: To show the potential improvement of Data Scalable codec in a streaming system, we built a packet-level simulator that takes in bandwidth trace and reference video as input, simulates the *end-to-end* video streaming process and then outputs decoded frames and end-to-end delay for each frame. We set round trip time to 200ms and uses drop-tail queue with length

Figure 6: *Sampled 150ms Frame Comparison (Brown Line: Bandwidth Drop; Red Line: Video Freeze; Red Dotted: Late frame)*

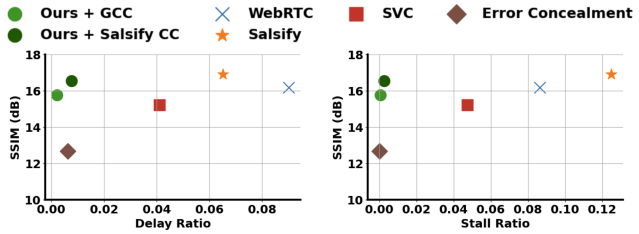


Figure 7: *End to End Simulation shows that Data Scalable codec could reduce video freeze with minor decrease in graphical quality.*

of 25 packet to simulate congestion packet loss. The traces are from eight LTE bandwidth traces in Mahimahi network-emulation tool [28].

4.2 Result

We measure the Data-Scalability of different codecs in Fig. 3 with fixed bit per pixel (bpp). Our codec with x% data arrival has similar quality with encoding by H264 in x% of the original bitrate (Oracle H264), which conforms closely to definition in §2.4. All other methods fail to provide such Data-Scalability. FEC performs better than Data Scalable codec if the loss rate is predicted closely, but performs worse most of time. SVC, Error Concealment, and Original Autoencoder [25] all have lower quality when random packet loss occurs.

We compare the video compression efficiency of our codec with state of art video codecs (H265 and H264) in Fig 4. Our codec has on par performance with H264 is also close to H265 under different bit per pixel (bpp). Comparing with H264 in terms of codec BDBR metric [7], our codec increases bitrate by 0.14% on FVC dataset and decreases bitrate by 12.9% on Cloud Gaming dataset, while H265 decreases bitrate by 34.6% and 39.4% respectively. Although our codec retains decent compression performance, Data Scalability comes at the cost of slightly worse quality than original Autoencoder.

In terms of speed. Our codec runs decently fast with a NVIDIA A40 GPU. On 360p, 480p, 720p videos, it is able to achieve 107.5fps, 74.6fps, 27.3fps respectively. We admit that this speed is indeed slower than traditional codecs [9] and is only workable with a high-quality GPU. The future speedup methods of Data Scalable codec is discussed in §5.

With Data Scalability, our codec enables better trade-off for real-time video streaming. Figure 7 compares system using our codec with other systems: Salsify [15] (Salsify CC + Loss frame skipping), WebRTC [18] (GCC + H264 + FEC), WebRTC + SVC, WebRTC + Error concealment. We measure average graphical quality (SSIM dB) and the ratio of delayed frames (over 400ms) and video stall (freeze over 200ms).

For baselines with comparable quality, our codec significantly decreases video delay and stall ratio by 19x and 14x respectively. This is because our system do not require retransmission during packet loss but substitute with moderate quality drop. This makes our average SSIM slightly lower than Salsify and WebRTC. But when comparing with error concealment that also has little stall, our solution achieves much better graphical quality.

We present a demo containing sampled frames from three decoded videos generated by different streaming logic on the same reference video during a bandwidth drop period from 6Mbps to 2Mbps. As in Fig 6, system using Data Scalable codec is able to decode with decent quality during packet loss, while others suffer from video freeze and late frames.

5 FUTURE DIRECTIONS

5.1 System Design

Data Scalability Training: Our paper shows a particular example that already improves Data-Scalablity. Nevertheless, this design could be improved in many ways.

The first question is *how to optimally set the loss distribution in training* to have better performance or shorter training time. We have shown in §4 that training autoencoders with fixed loss distribution already improves the Data Scalability under various loss rates. However, there could be smarter ways to set loss rate distribution. For example, we could gradually increase the percentage of higher loss rates (60%) to help codec learn to handle more loss progressively.

Another question is *how to better simulate the effects of real packet losses* during training. In §3, our training assumes that data losses each element in the coded tensor has the same probability to be zeroed. However, the autoencoder will then learn to handle all masking outcomes for the coded tensor. A better solution might be to re-couple packetization and training such that the codec has prior knowledge of which parts of the coded tensor will be put into a packet. This requires a new training pipeline but simplifies the task of the Data Scalable autoencoder.

Compute resource constraints: Although our benchmark on an NVIDIA A40 GPU shows promising result that Data Scalable codec could work in real-time, running the Data Scalable codecs with higher frame rate or on low-end devices requires more effort.

Besides general neural network speedup techniques including quantization and TensorRT, some other possible options include having "early exits" [26] in the neural network model to output a degraded frame when computing resources could not match the frame rate requirement, or specifically training fewer-layered decoder to adapt to lower-end receivers.

Bitrate control: Any video coding scheme must be capable of encoding a given video at a target bitrate set by bitrate adaptation logic. Recent autoencoders have supported variable output sizes [25], and our current implementation re-encodes the residual with another model if size differs by too much. However, this adds computational cost and do not fit closely with target bitrate. Future work is needed to fit the data size to bitrate more efficiently and precisely.

Protocol Design: Since Data Scalable coding has a fundamentally different utility curve, new streaming protocols are needed. Two questions that directly follow are:

(1) When does the decoder decide to decode the frame? In our simulator, our decoder decode when the first packet of the next frame arrive. However, in real networks, this may perform poorly due to unorderly arrival of the packets. Moreover, in low RTT networks, retransmission of the packets may give better quality without causing notable delay to users.

(2) How to synchronize the reference frames of the encoder and decoder after loss happens? On the decoder side, when we decoded frame f_i' with incomplete data and use it as the reference frame for the next frame f_{i+1}', it will be different from the perfect reference frame f_i from encoder side. The current solution is to send the packet loss information back to the encoder and let encoder updates its reference to replicate reference of decoder. However this method has a single-trip delay and causes extra compute overhead.

5.2 More Opportunities

Elastic Utility and Active Queue Management: Traditionally, utility of a flow only increases significantly when every packet arrived (*i.e.,* the "cliff" effect). This changes with data scalability that the utility now gradually increases with delivered packets, which is more elastic. This suggests that QoS AQM that maximizes flow completion will be less suitable with Data Scalable Codecs. Instead, early AQM schemes such as random early drop (RED) might become are adapt as they tend to deliver some packets for every flow.

Congestion Control: Congestion control aims to send data according to actual bandwidth. Some aggressive algorithms like Salsify CC [15] try to probe the bandwidth to have better utilization. However, when facing sudden bandwidth drop, aggressive algorithms often suffer from packet loss. Data-Scalable codec could minimize this damage by selectively probing bandwidth during more Data Scalable frames.

Moreover, when facing unstable links, some congestion control could result in extremely low utilization of the bandwidth [37]. With the ability to withstand packet loss, system with Data Scalable codec may not need to reduce sending rate even when bandwidth suddenly drops since it could increase back very soon.

Multimedia Streaming: Data Scalability could be applied to more types of multimedia streaming with the increasing power of neural network solutions [27, 39]. For example, smoothness is essential for VR and AR to provide immersive experience [8]; volumetric data consumes more bandwidth than 2D videos and thus has more optimization potential [39]. Adding Data Scalability modification to these applications could increases the system's ability to provide real-timeness robustly despite complex network conditions.

6 RELATED WORK

Neural Network for Videos: Recent work uses neural-network-based approach [13, 25, 33] to achieve on-par or even better compression efficiency than the traditional codecs. These networks can effectively capture the complex spatio-temporal patterns present in video data, enabling higher compression ratios while preserving high quality. There have also been advances in NN-based super resolution [20, 38] that leverage receiver-size computation power to achieve high-resolution video when bandwidth is limited. User only send lower resolution videos, which will be upscaled to higher resolution with receiver-side neural networks for better quality. This technique can be implemented on top of Data Scalable codec in low-bandwidth situation to get better quality together with loss-resilience.

Better Metric: Besides commonly used PSNR and SSIM, there have also been extensitve research on aligning objective metric with user experience. There is Frechet Inception Distance (FID) [16], which is usually used to access quality of generative image in computer vision that quantify the realism of image and LPIPS perceptual metric [40], which uses a learnt neural network to approximate human perceived similarity. There have also been metrics that try to directly calculate video quality including STRRED [1] and FVD [34].

Networking for Real Time Application: There have also been extensive research done recently across multiple network stacks to improve experience in video conferencing. Notably, L4S uses an Explicit Congestion Notification (ECN) scheme to signal router queue congestion to prevent loss and reduce delay [31]. GSO-Simulcast uses a centralized controller aware of more network conditions to control participants' video streams in multi-party conferencing [24]. Tambur uses streaming code to add FEC to a group of frames instead of a single frame to improve efficiency of traditional FEC [30]. Data Scalable codec could be combined with these methods to improve resiliency to packet loss with the cost of more computation.

7 CONCLUSION

This paper introduces Data Scalable codec for real-time video, whose key distinction is that its quality does not depend on complete data arrival but gradually increases with each new packet. We create the first Data Scalable codec by training neural autoencoders to perform under various loss rates. Admittedly, there remain challenges for such a system to be put into practice. We hope this paper sparks more discussion in broader community to fully exploit its potential.

ACKNOWLEDGEMENT

We thank the anonymous reviewers at EMS, Junchen Jiang, Xu Zhang, and Yuanhang Su for their feedback and suggestions.

REFERENCES

[1] 2013. ST-RRED Video Quality Predictor. http://live.ece.utexas.edu/research/quality/ST-RRED. (2013).

[2] 2020. UVG Dataset: 50/120fps 4K sequences for video codec analysis and development. https://ultravideo.fi/. (2020).

[3] 2020. WebRTC Media Servers. https://webrtc.ventures/2020/12/webrtc-media-servers-sfus-vs-mcus/. (2020).

[4] 2021. Cloud-Gaming-Video-Dataset . https://github.com/stootaghaj/Cloud-Gaming-Video-Dataset. (2021).

[5] 2022. FVC Workshop. https://fvc-workshop.github.io/. (2022).

[6] 2023. Amazon Mechanical Turk. https://www.mturk.com/. (2023).

[7] 2023. Bjøntegaard-Delta Interpolation. https://github.com/FAU-LMS/bjontegaard. (2023).

[8] 2023. Features of WebRTC VR Streaming. https://flashphoner.com/features-of-webrtc-vr-streaming/. (2023).

[9] 2023. Handbrake Codec Performance. https://handbrake.fr/docs/en/latest/technical/performance.html. (2023).

[10] 2023. WebRTC. https://webrtc.org/. (2023).

[11] Claudio Alberti, Daniele Renzi, Christian Timmerer, Christopher Mueller, Stefan Lederer, Stefano Battista, and Marco Mattavelli. 2013. Automated QoE evaluation of Dynamic Adaptive Streaming over HTTP. In *2013 Fifth International Workshop on Quality of Multimedia Experience (QoMEX)*. 58–63. https://doi.org/10.1109/QoMEX.2013.6603211

[12] Nabajeet Barman and Maria G. Martini. 2019. QoE Modeling for HTTP Adaptive Video Streaming–A Survey and Open Challenges. *IEEE Access* 7 (2019), 30831–30859. https://doi.org/10.1109/ACCESS.2019.2901778

[13] Mallesham Dasari, Kumara Kahatapitiya, Samir R. Das, Aruna Balasubramanian, and Dimitris Samaras. 2022. Swift: Adaptive Video Streaming with Layered Neural Codecs. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*. USENIX Association, Renton, WA, 103–118. https://www.usenix.org/conference/nsdi22/presentation/dasari

[14] Zhengfang Duanmu, Kai Zeng, Kede Ma, Abdul Rehman, and Zhou Wang. 2017. A Quality-of-Experience Index for Streaming Video. *IEEE Journal of Selected Topics in Signal Processing* 11, 1 (2017), 154–166. https://doi.org/10.1109/JSTSP.2016.2608329

[15] Sadjad Fouladi, John Emmons, Emre Orbay, Catherine Wu, Riad S Wahby, and Keith Winstein. 2018. Salsify:{Low-Latency} Network Video through Tighter Integration between a Video Codec and a Transport Protocol. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*. 267–282.

[16] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. 2017. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/8a1d694707eb0fefe65871369074926d-Paper.pdf

[17] Tobias Hossfeld, Raimund Schatz, Ernst Biersack, and Louis Plissonneau. 2013. *Internet Video Delivery in YouTube: From Traffic Measurements to Quality of Experience*. Vol. 7754. 264–301. https://doi.org/10.1007/978-3-642-36784-7_11

[18] Bart Jansen, Timothy Goodwin, Varun Gupta, Fernando Kuipers, and Gil Zussman. 2018. Performance Evaluation of WebRTC-Based Video Conferencing. *SIGMETRICS Perform. Eval. Rev.* 45, 3 (mar 2018), 56–68. https://doi.org/10.1145/3199524.3199534

[19] Jaeyeon Kang, Seoung Wug Oh, and Seon Joo Kim. 2022. Error Compensation Framework for Flow-Guided Video Inpainting. (2022). arXiv:cs.CV/2207.10391

[20] Jaehong Kim, Youngmok Jung, Hyunho Yeo, Juncheol Ye, and Dongsu Han. 2020. Neural-Enhanced Live Streaming: Improving Live Video Ingest via Online Learning. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM '20)*. Association for Computing Machinery, New York, NY, USA, 107–125. https://doi.org/10.1145/3387514.3405856

[21] Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. arXiv:http://arxiv.org/abs/1312.6114v10

[22] Sunil Kumar, Liyang Xu, Mrinal K Mandal, and Sethuraman Panchanathan. 2006. Error resiliency schemes in H. 264/AVC standard. *Journal of Visual Communication and Image Representation* 17, 2 (2006), 425–450.

[23] Weiping Li. 2001. Overview of fine granularity scalability in MPEG-4 video standard. *IEEE Transactions on circuits and systems for video technology* 11, 3 (2001), 301–317.

[24] Xianshang Lin, Yunfei Ma, Junshao Zhang, Yao Cui, Jing Li, Shi Bai, Ziyue Zhang, Dennis Cai, Hongqiang Harry Liu, and Ming Zhang. 2022. GSO-Simulcast: Global Stream Orchestration in Simulcast Video Conferencing Systems. In *Proceedings of the ACM SIGCOMM 2022 Conference (SIGCOMM '22)*. Association for Computing Machinery, New York, NY, USA, 826–839. https://doi.org/10.1145/3544216.3544228

[25] Guo Lu, Wanli Ouyang, Dong Xu, Xiaoyun Zhang, Chunlei Cai, and Zhiyong Gao. 2019. DVC: An End-to-end Deep Video Compression Framework. (2019). arXiv:eess.IV/1812.00101

[26] Yoshitomo Matsubara, Marco Levorato, and Francesco Restuccia. 2022. Split Computing and Early Exiting for Deep Learning Applications: Survey and Research Challenges. *ACM Comput. Surv.* 55, 5, Article 90 (dec 2022), 30 pages. https://doi.org/10.1145/3527155

[27] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. *CoRR* abs/2201.05989 (2022). arXiv:2201.05989 https://arxiv.org/abs/2201.05989

[28] Ravi Netravali, Anirudh Sivaraman, Somak Das, Ameesh Goyal, Keith Winstein, James Mickens, and Hari Balakrishnan. 2015. Mahimahi: Accurate {Record-and-Replay} for {HTTP}. In *2015 USENIX Annual Technical Conference (USENIX ATC 15)*. 417–429.

[29] Alexander Raake, Marie-Neige Garcia, Werner Robitza, Peter List, Steve Göring, and Bernhard Feiten. 2017. A bitstream-based, scalable video-quality model for HTTP adaptive streaming: ITU-T P.1203.1. In *Ninth International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, Erfurt. https://doi.org/10.1109/QoMEX.2017.7965631

[30] Michael Rudow, Francis Y Yan, Abhishek Kumar, Ganesh Ananthanarayanan, Martin Ellis, and KV Rashmi. 2023. Tambur: Efficient loss recovery for videoconferencing via streaming codes. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. 953–971.

[31] Koen De Schepper and Bob Briscoe. 2023. The Explicit Congestion Notification (ECN) Protocol for Low Latency, Low Loss, and Scalable Throughput (L4S). RFC 9331. (Jan. 2023). https://doi.org/10.17487/RFC9331

[32] Anika Seufert, Florian Wamser, David Yarish, Hunter Macdonald, and Tobias Hoßfeld. 2021. QoE Models in the Wild: Comparing Video QoE Models Using a Crowdsourced Data Set. In *2021 13th International Conference on Quality of Multimedia Experience (QoMEX)*. 55–60. https://doi.org/10.1109/QoMEX51781.2021.9465422

[33] Vibhaalakshmi Sivaraman, Pantea Karimi, Vedantha Venkatapathy, Mehrdad Khani, Sadjad Fouladi, Mohammad Alizadeh, Frédo Durand, and Vivienne Sze. 2023. Gemino: Practical and Robust Neural Compression for Video Conferencing. (2023). arXiv:cs.NI/2209.10507

[34] Thomas Unterthiner, Sjoerd van Steenkiste, Karol Kurach, Raphaël Marinier, Marcin Michalski, and Sylvain Gelly. 2018. Towards Accurate Generative Models of Video: A New Metric & Challenges. *CoRR* abs/1812.01717 (2018). arXiv:1812.01717 http://arxiv.org/abs/1812.01717

[35] Nikolas Wehner, Anika Seufert, Tobias Hoßfeld, and Michael Seufert. 2023. Explainable Data-Driven QoE Modelling with XAI. In *2023 15th International Conference on Quality of Multimedia Experience (QoMEX)*. 7–12. https://doi.org/10.1109/QoMEX58391.2023.10178499

[36] Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T Freeman. 2019. Video enhancement with task-oriented flow. *International Journal of Computer Vision* 127, 8 (2019), 1106–1125.

[37] Francis Y. Yan, Jestin Ma, Greg D. Hill, Deepti Raghavan, Riad S. Wahby, Philip Levis, and Keith Winstein. 2018. Pantheon: the training ground for Internet congestion-control research. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*. USENIX Association, Boston, MA, 731–743. https://www.usenix.org/conference/atc18/presentation/yan-francis

[38] Hyunho Yeo, Youngmok Jung, Jaehong Kim, Jinwoo Shin, and Dongsu Han. 2018. Neural Adaptive Content-aware Internet Video Delivery. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*. USENIX Association, Carlsbad, CA, 645–661. https://www.usenix.org/conference/osdi18/presentation/yeo

[39] Anlan Zhang, Chendong Wang, Bo Han, and Feng Qian. 2022. YuZu: Neural-Enhanced Volumetric Video Streaming. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*. USENIX Association, Renton, WA, 137–154. https://www.usenix.org/conference/nsdi22/presentation/zhang-anlan

[40] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. 2018. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *CVPR*.